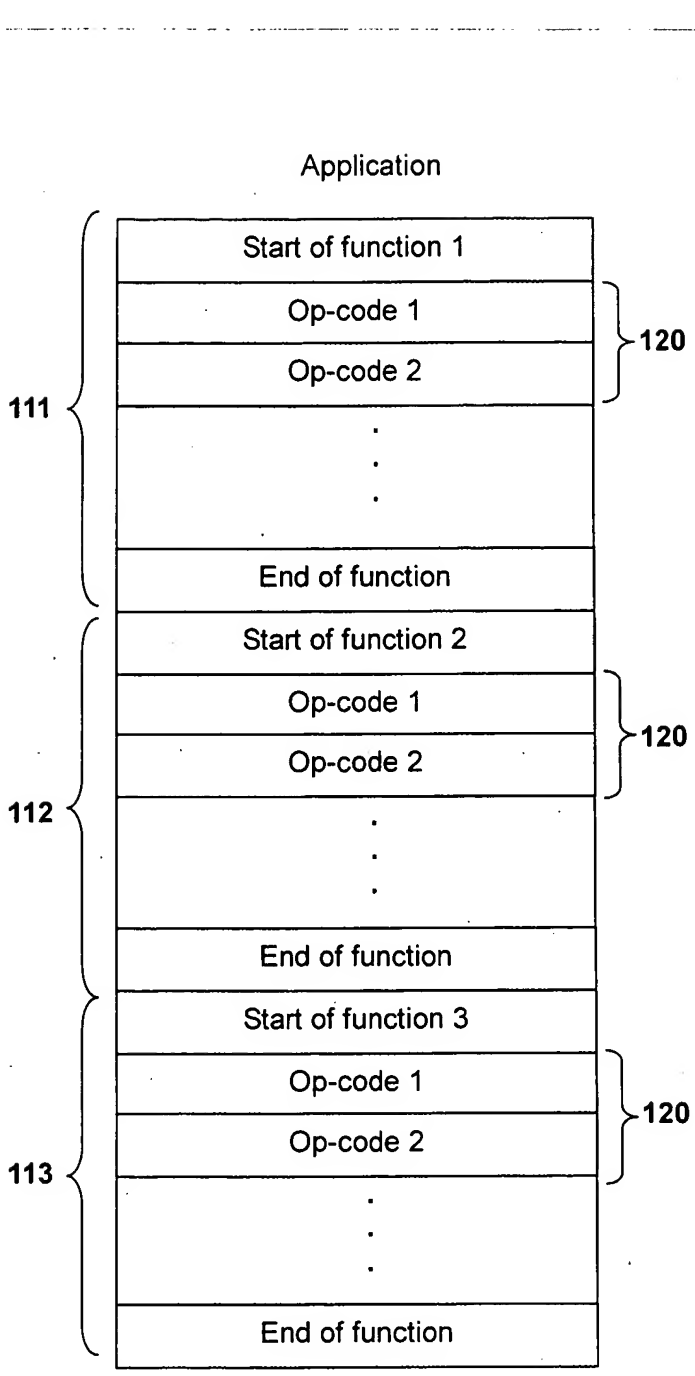
100
Application
Start of function 1
Op-code 1
Op-code 2
120
111
End of function
Start of function 2
Op-code 1
Op-code 2
120
112
End of function
Start of function 3
Op-code 1
Op-code 2
120
113
End of function

Fig. 1

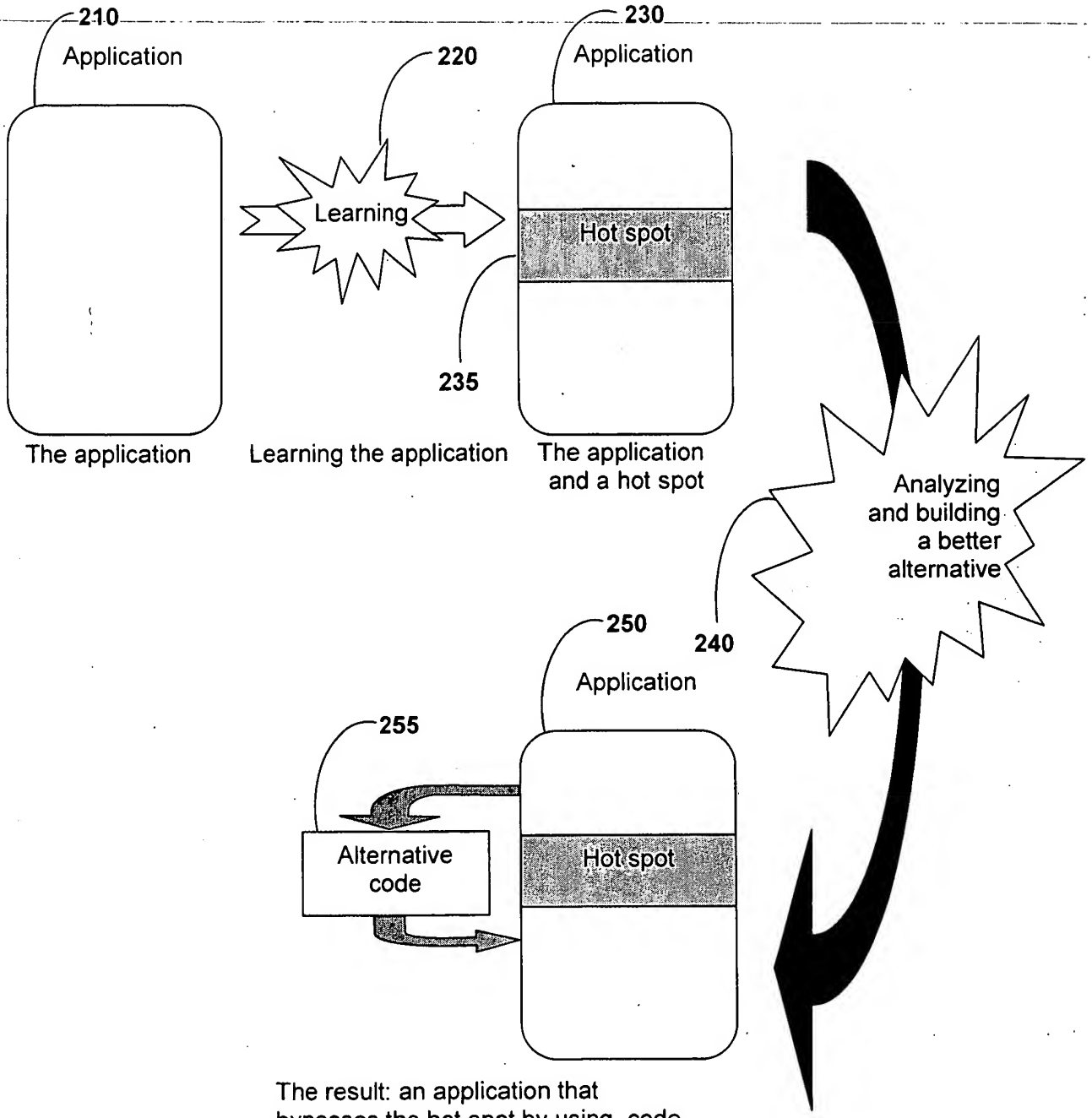
210
Application
220
230
Application
Learning
Hot spot
235
The application
Learning the application
The application and a hot spot
Analyzing and building a better alternative
240
250
Application
255
Alternative code
Hot spot
The result: an application that bypasses the hot spot by using code that was costumized and optimized to the specific hardware.

Fig. 2

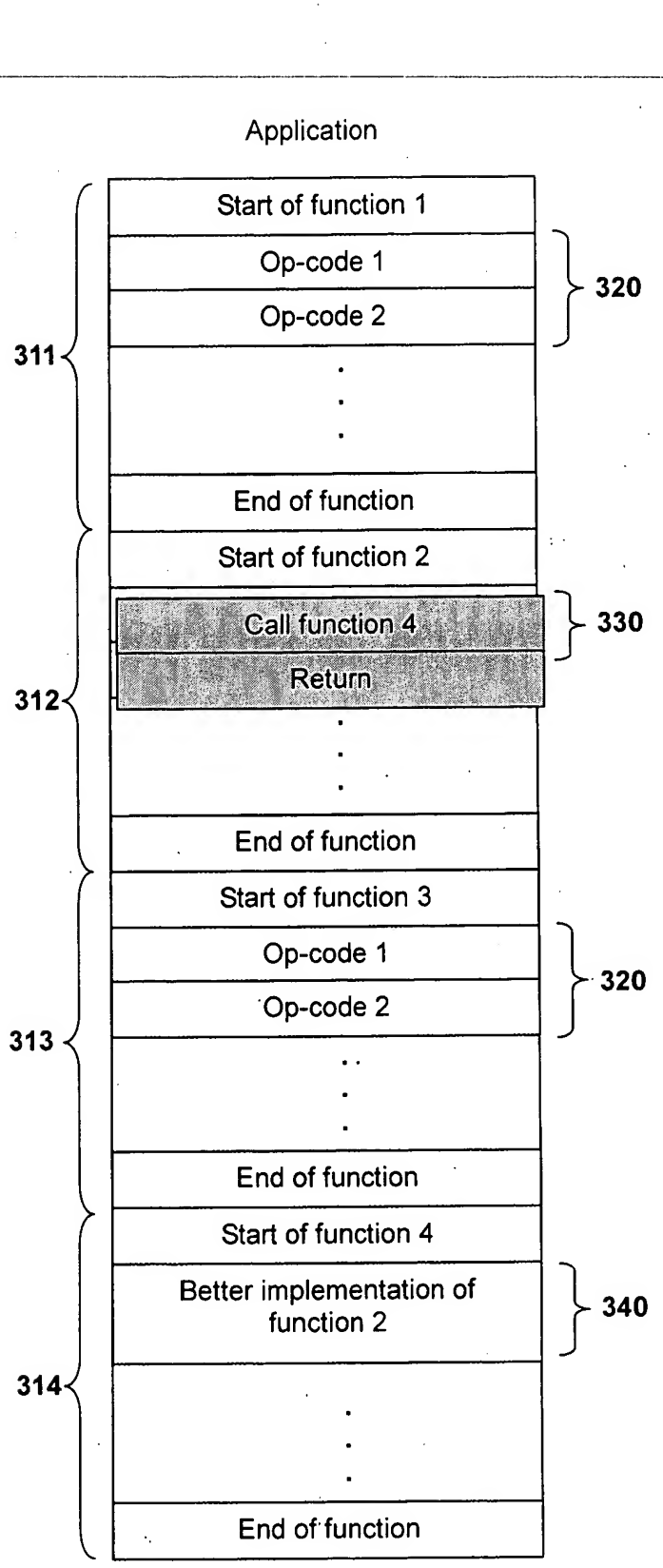
300
Application
Start of function 1
Op-code 1
Op-code 2
320
311
End of function
Start of function 2
Call function 4
330
Return
312
End of function
Start of function 3
Op-code 1
Op-code 2
320
313
End of function
Start of function 4
Better implementation of function 2
340
314
End of function

Fig. 3

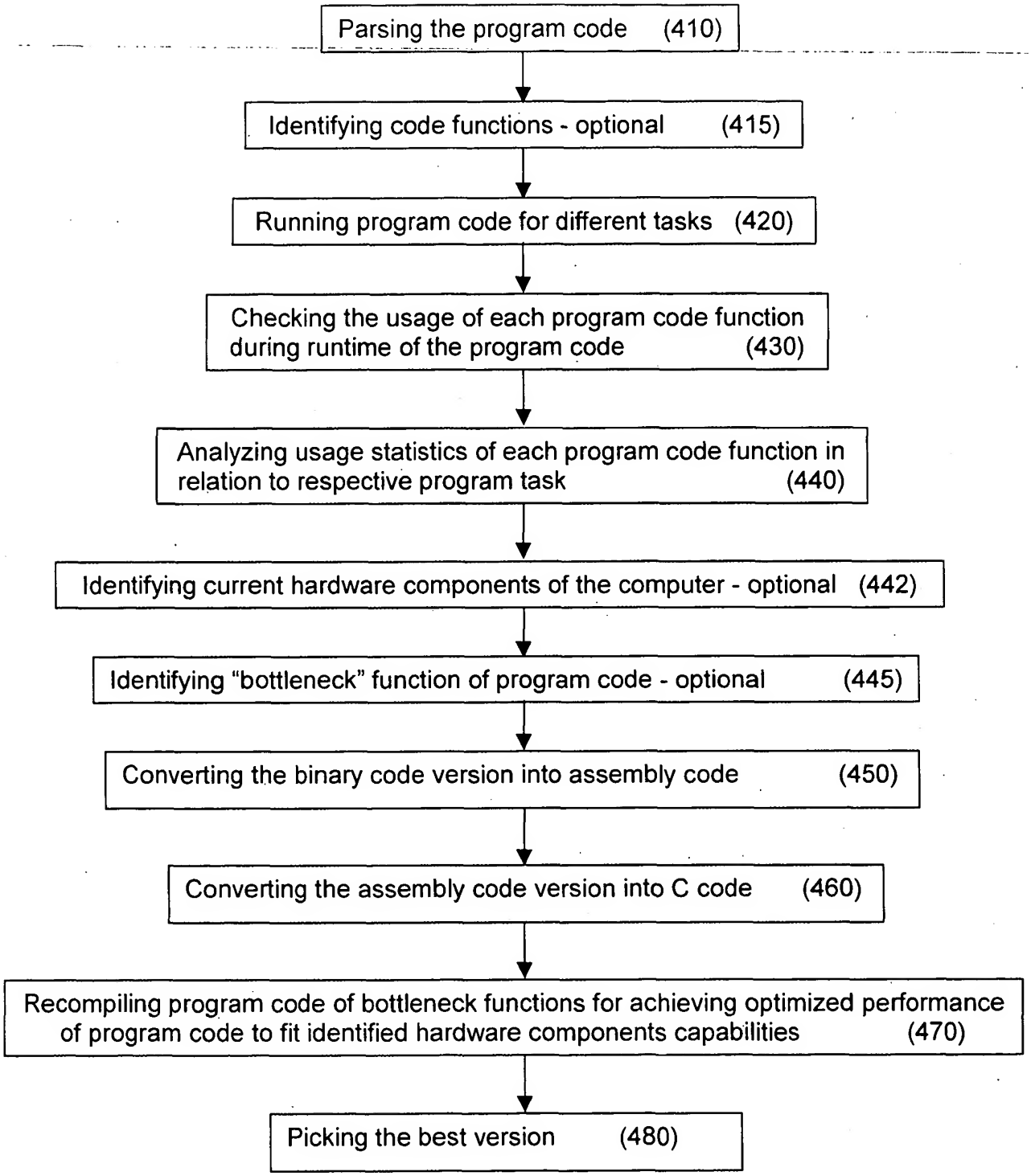
Parsing the program code (410)
Identifying code functions - optional (415)
Running program code for different tasks (420)
Checking the usage of each program code function during runtime of the program code (430)
Analyzing usage statistics of each program code function in relation to respective program task (440)
Identifying current hardware components of the computer - optional (442)
Identifying "bottleneck" function of program code - optional (445)
Converting the binary code version into assembly code (450)
Converting the assembly code version into C code (460)
Recompiling program code of bottleneck functions for achieving optimized performance of program code to fit identified hardware components capabilities (470)
Picking the best version (480)

Fig. 4